Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Leo Dong, Alex Johnson, Janani Kannan, Ashwin Wariar

University of Minnesota, Twin Cities

February 2024

Table of Contents

Table of Contents (contd.)

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Janani Kannan	1/31	Import template	0.00
Alex Johnson	2/5	1.1 Draft Purpose	0.05
Team 13	2/7	1 Draft full section, sources, check completeness	0.07
Janani Kannan	2/9	2.1 - 2.3 Draft system overview	0.09
Ashwin Wariar	2/10	3.2 - 3.3 Draft interfaces	0.10.a
Janani Kannan	2/10	2.1 - 2.5 4 Notes on System Features	0.10.b
Alex Johnson	2/11	2.2 - 3.5 Edits and drafting 5 Notes on nonfunctional requirements	0.11.a
Ashwin Wariar	2/11	5 Additional notes on nonfunctional requirements	0.11.b
Leo Dong, Janani Kannan, Ashwin Wariar	2/11	4 Draft system features	0.11.c
Janani Kannan	2/11	1 - 2 Remove templates, Appendix A	0.11.d
Alex Johnson	2/11	5 Draft nonfunctional requirements	0.11.e
Alex Johnson	2/12	2.7 - 3.1 Assumptions and user interface, Appendix A	0.12.a
Leo Dong	2/12	4 System features	0.12.b
Leo Dong, Alex Johnson	2/12	6 Other / legal requirements	0.12.c
Leo Dong	2/12	4 System features	0.12.d
Team 13	2/12	Finalize the SRS document	1.0

1. Introduction

1.1 Purpose

This document provides a detailed description of a standalone software for an aggregated voting system. This document is relevant to version 1.0 of the software. This software calculates the results of two types of voting systems: Open Party Listing and Closed Party Listing. The functional and nonfunctional aspects of the Voting System software are described in detail. This document is intended for users and developers, allowing for a smoother user experience as well as aiding in testing, extending, and maintaining this software.

1.2 Document Conventions

Each section starts broadly before breaking into a more granular level. This document follows the IEEE Software Requirements and Specifications document conventions. Priority is assumed to be inherited from parent clauses.

Typographic conventions:
- Font style: Times New Roman is used throughout the document
- Headings: 18pt font, numbered, bold
- Subheadings: 14pt font, numbered, bold
- Body text: 11pt font, labels in bold

Comments and annotations: In some sections, additional notes are written in brackets: [...]. These are meant to keep track of a broad overview of ideas, and provide a summary of any content that follows.

Use case conventions: Use cases are numbered, and follow the format of UC_<use case number>

Labels in Section 4: For each System Feature, the user actions, system actions, and system responses listed under 4.x.2 are numbered. User action labels and system action labels are highlighted in different colors for clarity, following this format:
- **User-Action-<action #>**
- **System-Action-<action #>**

1.3 Intended Audience and Reading Suggestions

This document is for users; election officials, testers, and developers. This document is organized as follows: an overall description of the software describing the product and its functionality; design constraints and other non-functional requirements, bounding the system under client requirements and local regulations; and assumptions and dependencies, where the client did not provide rigid guidance or where external systems are used. External interface requirements are described in detail, as well as system features that describe the functional requirements required for this product.

1.4 Product Scope

This system determines and displays the results of elections that use one of two types of party list voting systems: Closed Party List and Open Party List. In a Closed Party List system, voters choose a party as a whole. Each party has a predetermined order of candidates, and individual candidate selections are made by the party. In an Open Party List system, voters elect specific candidates. See §1.5 or §6 for more information. Once votes are calculated depending on the type of voting system, results are displayed to the end user. This software provides an efficient mechanism for election officials to obtain the results of an election by following a set of simple steps. If implemented, this could help businesses and any other parties involved reduce costs associated with processing election results.

1.5 References

[Website with information about OPL and CPL]
FairVote. (2022, June 22). *Proportional Representation Voting Systems*. FairVote.
https://fairvote.org/archives/ proportional-representation-voting-systems/

[Permalinks to our use cases document, the template used to write use cases, and instructions for this part of the project; all of these are available on our shared Github repository]
Use Cases Document
[https://github.umn.edu/umn-csci-5801-02-S24/repo-Team13/blob/1dcd88a368bc55640d9651d9673fa3f7547d7f60/p1/5801%20Use%20Cases.docx]
Use Case Template
[https://github.umn.edu/umn-csci-5801-02-S24/repo-Team13/blob/1dcd88a368bc55640d9651d9673fa3f7547d7f60/p1/UseCaseTemplate.jpg]
Voting Systems SRS Instructions
[https://github.umn.edu/umn-csci-5801-02-S24/repo-Team13/blob/c4211ad9df07f162f2f98df89e3fe150c432eca3/p1/Project1_Waterfall_VotingSRS_Spring2024.pdf]

2. Overall Description

2.1 Product Perspective

Our voting system is a new, standalone product. Using a single program that performs computations based on a CSV file containing ballot information, our system calculates and presents the results of either an Open Party Listing or a Closed Party Listing based election. This system can be run on any UMN CSE Labs machine. At the top level, our system has three major components: user interface, file processing, and results computation. An overview of these system parts is shown in Figure 1 below:

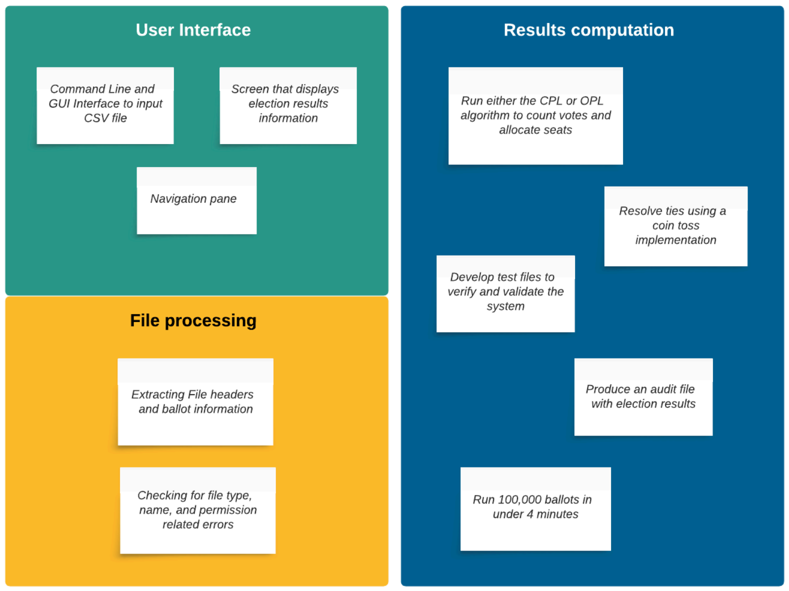
User Interface
Command Line and GUI Interface to input CSV file
Screen that displays election results information
Navigation pane
File processing
Extracting File headers and ballot information
Checking for file type, name, and permission related errors
Results computation
Run either the CPL or OPL algorithm to count votes and allocate seats
Resolve ties using a coin toss implementation
Develop test files to verify and validate the system
Produce an audit file with election results
Run 100,000 ballots in under 4 minutes

Fig. 1: Major system components with brief description

2.2 Product Functions

- An interface that accepts a CSV file as an input
- A single program to compute the results of either an OPL or a CPL election
- Ability to view the results of the election including relevant information about candidates and seat allocations
- An audit file to test correctness

2.3 User Classes and Characteristics

This system is designed for three classes of users: developers, election officials, and media personnel.
Programmers and testers: developers building the software and testers auditing the system have the highest level of expertise in understanding system functionality. They are primarily responsible for ensuring the correctness, accuracy, and reliability of the software.
Election officials: the target user base tasked with determining the result of an election, holding the highest security clearance, providing the raw vote CSV files, and specifying the system requirements.
Media personnel: the lowest technical expertise and security clearance, disseminate the election results.

2.4 Operating Environment

Our system is equipped to run on Linux operating systems, such as UMN CSE Labs machines. An updated distribution of Ubuntu is recommended. The program and test files are written in C++. Any IDE that supports C++ should be able to support software modifications and testing.

2.5 Design and Implementation Constraints

For all software components, C++ is used. Developers who wish to make changes to program files need an IDE that supports this language. The program is also limited to processing a single CSV file (per election cycle or program run) containing ballot information in a highly specific format. It is therefore the responsibility of the testers and election officials to ensure that no errors are present in the CSV file. Election officials must further guarantee the veracity of the election information for the system to produce the correct result. The environment in which this system is run must be capable of evaluating 100,000 ballots in under four minutes on a single thread. The end user is responsible for enforcing additional security policies.

2.6 User Documentation

A ReadMe is provided, detailing the OPL and CPL algorithms, error messages, and usage examples. The ReadMe also details how to compile this system and required nonstandard external libraries. Documentation is provided in html format as created by Doxygen.

2.7 Assumptions and Dependencies

It is assumed that the environment is capable of locking the CSV file to prevent malicious behavior while the system is run and has sufficient memory to store and process the file. This file is in the same directory as the system instance. The environment is assumed to be an up-to-date UMN CSE Labs machine or equivalent. This system assumes this CSV is valid, so it is the responsibility of the user to verify the correctness of the submitted file. Only one file is to be submitted per run.

3. External Interface Requirements

3.1 User Interfaces

Users interact with the system in two ways: command line and graphical interface. When launched from the command line, the system interprets the first argument as the CSV file. A missing or invalid argument opens a GUI and prints a help message, continuing to prompt the user until canceled or a valid file is found. The terminal interface is intended to allow for automation and testing, while a GUI caters to less technical users. The GUI may also be more convenient, allowing users to select a file rather than typing a file's name.
This GUI features a submit or select button, a cancellation button, and a help button. Pressing `Esc` at this stage terminates the program, while `Enter` submits the chosen file, or prompts the user until a valid file is found. This GUI is not to occupy more than one-quarter of the screen. Whenever a second prompt is required, such as when entering an invalid file on the command line or GUI, a help message is displayed. This message is also displayed when the `Help` button is pressed. The help message is not initially displayed when the GUI is opened as a result of a missing command line file name. Other error messages are displayed to the terminal with a stack trace and brief explanation. This GUI is only needed when selecting the input file.

3.2 Hardware Interfaces

The program is coded in C++. Therefore, the hardware must be capable of compiling and running C++ code. Input files of election data must be read and processed, so the hardware must possess sufficient memory and computational power to store and process the data in a reasonable time. Hardware communication is managed by the operating system.

3.3 Software Interfaces

The environment must be able to compile and execute C++ code. This means that the environment should possess a compiler that matches the needs of the code, as well as the standard C++ libraries that are required for the program to run successfully. Specialized libraries are not required. The operating system must handle hardware communication, while no interprocess or network communication is required.

3.4 Communications Interfaces

For communications interfaces, the system must be able to access the election voting data from the same directory. Interprocess and network communication is not required. For security, the environment must be capable of locking a file to prevent malicious writing. Synchronization is not required as the provided CSV file must be fully written before being consumed by the system.

4. System Features

4.1 Audit File Output

4.1.1 Description and Priority

Description: The system produces an audit file with information about the election, including the type of election (CPL or OPL), number of parties, number of ballots, number of seats, candidate names and party affiliation, calculations for the largest remainder approach, seat allotments, winner information, and their party affiliation.

Priority: This has a benefit priority between 7.5 - 8. This falls between medium and high priority. System feature 4.2 allows users to view election results. Some users may only want to view results and may not have the necessity to save a file with the information for later access.

4.1.2 Stimulus / Response Sequences

User-Action-1: The user provides a CSV file with ballot information by entering a desired input file name through the command line or prompt (see System Feature 4.3).

System-Action-1: The system processes the CSV file, computes seat allocations, and determines winners (see System Features 4.4, 4.5, 4.6, 4.7, and 4.8).

Response-1: The system produces an audit file containing all information.

Response-2: A system failure occurs during User-Action-1 or System-Action-1, and the system exits with the corresponding error message.

4.1.3 Functional Requirements

REQ-1: System feature 4.3 - File Name must function correctly and reliably.
REQ-2: System Feature 4.4 - Process CSV File must function correctly and reliably.
REQ-3: System Features 4.5, 4.6, 4.7, and 4.8 must function correctly and reliably to compute election results.
REQ-4: The system must produce an audit file with the following information (listed as part of **UC_001** in our use case document):

- The type of election (CPL or OPL)
- Number of Parties
- Number of Ballots
- Number of Seats
- The Candidates' Names and Party Affiliation
- Calculation for the largest remainder approach
- A table representing seat allotments
- Seat Winners and their Party Affiliation
- OPL has votes received

REQ-5: The system shall produce the correct audit file with an intuitive file name matching with the input CSV file.

REQ-6: The system shall allow users to access the audit file by providing a download button.

4.2 Display Election Information

4.2.1 Description and Priority

Description: The system displays the election winners on the screen with the necessary information, such as the number of ballots cast and the statistics for everyone, including those that did not win, like number of votes received, percentage of votes received, etc.

Priority: It has a high benefit priority of 9, allowing users to see the election results immediately after the system's execution. It provides an easy and convenient user experience as the system's primary purpose is to calculate the input election votes and determine the winner by displaying them on screen. System feature 4.1 allows users to save and download the information as an audit file.

4.2.2 Stimulus / Response Sequences

User Action-1: The user provides a CSV file with ballot information.

System Action-1: The system processes the CSV file, computes seat allocations, and determines winners (see System Features 4.4, 4.5, 4.6, 4.7, and 4.8).

Response-1: The system presents all relevant information to the screen.

User-Action-2: The user provides an invalid file that caused any functional failure during the program's execution.

Response-2: The system exits with the corresponding error message.

4.2.3 Functional Requirements

REQ-1: System Feature 4.4 - Process CSV File must function correctly and reliably.

REQ-2: System Features 4.5, 4.6, 4.7, and 4.8 must function correctly and reliably to compute election results.

REQ-3: The system shall display the election result to the user in GUI.

REQ-4: The system shall allow the user to re-execute the whole process.

4.3 Obtain File Name

4.3.1 Description and Priority

Description: The system obtains the file names through two different mechanisms: 1) the user enters the file name as an argument on the command line before running the application; 2) The user is prompted to enter the file name while the application is running.

Priority: This has a penalty priority of 7. This falls between median and high priority because it provides the user with the flexibility to enter the target file name into the system via two different methods. Without this feature, the user would have trouble accessing the wanted file, and the whole election result computation process would be stalled.

4.3.2 Stimulus / Response Sequences

User-Action-1: When starting the application, the user has the choice to input the file name on the command line. If so, System Response-1 arises; otherwise, it proceeds to Action-2.

Response-1: The system checks whether the file name provided on the command line is valid. The system exits with an error message if the file is not found or an invalid file name is entered.

Response-2: The file name is valid, and the system proceeds to System Feature 4.4 Process CSV File.

System-Action-1: The system prompts users for a filename if the application has started without an input file name.

User-Action-2: Users enter the file name through the prompt.

Response-3: The system checks whether the file name provided in the prompt is valid. The system shows an error message and asks the user to enter again if the file is not found or an invalid file name is entered.

Response-4: The file name provided is valid, and the system proceeds to System Feature 4.4 Process CSV File.

User-Action-3: Users fail to enter the file name through the prompt in User-Action-2.

Response-5: The system stays on hold or exits with an error message after exceeding the limited waiting time.

User-Action-4: Users cancel to enter the file name through the prompt in User-Action-2.

Response-6: The system terminates without error.

4.3.3 Functional Requirements

REQ-1: The system shall display the raised error message in the user interface.
REQ-2: The system shall ask the user to enter (or re-enter) the file name through the prompt.
REQ-3: The system shall provide the user a help button to explain further clarified instructions for selecting a file or entering a file name.

4.4 Process CSV File

4.4.1 Description and Priority

Description: The system first opens a preprocessed CSV file, which stores valid ballots. Then, the system reads the first line of the input CSV file to determine whether the election ballot is for Open Party List or Closed Party List voting. Based on different voting, the system processes the file by extracting all critical information needed directly from the file to execute the result computation algorithm. Ultimately, the system closes the file without any modifications or security issues.

Priority: This has a penalty priority of 8 and a cost priority of 6. It contains the preliminary steps that set up the foundational information for the proceeding computation of results. The feature provides the automation of information extraction to the software without any additional system or user input. In addition, this internal feature saves the cost of developing or utilizing an external information extraction system.

4.4.2 Stimulus / Response Sequences

User-Action-1: The user enters a desired input CSV file name through the command line or prompt (see System Feature 4.3).

Response-1: A system failure occurs during User-Action-1, and the system exits with the corresponding error message

System-Action-1: The system searches the file from the designated directory.

Response-2: If the required file is inaccessible, the system exits with an error message; otherwise, it is processed to System-Action-2.

System-Action-2: The system reads the first line of the file to identify the voting type (either CPL or OPL).

Response-3: The system checks whether the voting type is missing or an alternative voting type is detected. If so, the system exits with an error message; otherwise, it is processed to System-Action-3.

System-Action-3: The system locates and extracts all required information, such as seats, ballots, and parties/candidates, based on the voting type.

Response-4: If the system cannot locate any required information, an invalid election vote file is detected and exits with an error message.

Response-5: The system completes the extraction process, closes the file, and proceeds to the result computation as no error occurs.

4.4.3 Functional Requirements

REQ-1: System Feature 4.3 - Obtain File Name must function correctly and reliably.
REQ-2: The system shall obtain a correct and robust OPL information extraction program.
REQ-3: The system shall obtain a correct and robust CPL information extraction program.
REQ-4: The system shall display the raised error message in the user interface.
REQ-5: The system must maintain the same information and authentication of the file data from opening to closing of the file.

4.5 Resolve Ties

4.5.1 Description and Priority

Description: The system resolves a tie in the election result by simulating a virtual coin toss to determine the winner. It is critical that when an election ends in a tie, users expect an objective solution to the result without subject bias. The whole tiebreaker determination process does not involve any user interactions.
Priority: This feature has a benefit priority of 3. This feature is not one of the essential project objectives in this software development, but it is considered significantly practical to provide the user the convenience of skipping the process of manually resolving the tiebreaker.

4.5.2 Stimulus / Response Sequences

User-Action-1: The user enters a desired input CSV file name through the command line or prompt (see System Feature 4.3).

System-Action-1: The system processes relevant information (see System Feature 4.4) and computes an election result through the CPL or OPL Algorithm (see System Features 4.6 and 4.7), and the result ends where the top two or more candidates receive the same number of votes.

Response-1: The system labels tied candidates and simulates the coin toss to select the final winner. It then proceeds to System Feature 4.1 and System Feature 4.2.

System-Action-2: The system computes the result smoothly without any tiebreakers.

Response-2: The system ignores this feature and proceeds to System Feature 4.1 and System Feature 4.2.

4.5.3 Functional Requirements

REQ-1: System Feature 4.3 - Obtain File Name must function correctly and reliably.
REQ-2: System Feature 4.4 - Process CSV File must function correctly and reliably.
REQ-3: System Features 4.5, 4.6, 4.7, and 4.8 must function correctly and reliably to compute election results.
REQ-4: The simulation probability of flipping a coin for heads or tails must be fair (50%) and cryptographically secure.

4.6 CPL Algorithm

4.6.1 Description and Priority

Description: The system calculates the election results given ballot information using the Closed Party Listing algorithm. The seat allocation process is completed afterward without any direct user access.
Priority: This feature has a high penalty priority of 9. It contains the detailed steps of how the system computes an election result in the Closed Party Listing style. Implementing this algorithm provides a crucial solution to one of the problems the software is designed to solve.

4.6.2 Stimulus / Response Sequences

User-Action-1: The user enters a desired input CSV file name through the command line or prompt (see System Feature 4.3).

System-Action-1: The system processes the information from the ballots in the Closed Party Listing style (see System Feature 4.4). The system identifies the voter's party choice and increments the number of votes for the party by one.

System-Action-2: The system calculates seat allocations using the largest remainder approach (see Reference FairVote). If a tie occurs, the system resolves a tie using a coin flip simulation (see System Feature 4.5- Resolve Ties).

Response-1: The system determines the ranking of parties based on the seats allocated.

Response-2: The system determines the candidates receiving the seats and stores information about the list of ranked candidates and their affiliated party.

Response-3: The system stores all information in the appropriate data structure and proceeds to System Feature 4.1 Audit File Output and System Feature 4.2 Display Election Information.

Response-4: A system failure occurs during User-Action-1 to System-Action-2, and the system exits with the corresponding error message.

4.6.3 Functional Requirements

REQ-1: System Feature 4.3 - Obtain File Name must function correctly and reliably.
REQ-2: System Feature 4.4 - Process CSV File must function correctly and reliably.
REQ-3: System Feature 4.5 - Resolve Ties must function correctly and reliably.
REQ-4: The system must correctly and robustly implement the largest reminder approach.

4.7 OPL Algorithm

4.7.1 Description and Priority

Description: The system calculates the election results given ballot information using the Open Party Listing algorithm. The seat allocation process is completed afterward without any direct user access.
Priority: This feature has a high penalty priority of 9. It contains the detailed steps of how the system computes an election result in an Open Party Listing style. Implementing this algorithm provides a crucial solution to one of the problems the software is designed to solve.

4.7.2 Stimulus / Response Sequences

User-Action-1: The user enters a desired input CSV file name through the command line or prompt (see System Feature 4.3).

System-Action-1: The system processes the information from the ballots in the Closed Party Listing style (see System Feature 4.4).The system identifies the voter's candidate choice and increments the number of votes for the candidate by one and the number of votes for the party of the candidate by one.

System-Action-2: The system calculates seat allocations using the largest remainder approach (see Reference FairVote). If a tie occurs, the system resolves a tie using a coin flip simulation (see System Feature 4.5- Resolve Ties).

Response-1: The system determines the ranking of candidates based on the seats allocated.

Response-2: The system determines the candidates receiving the seats and stores information about the list of ranked candidates and their affiliated party.

Response-3: The system stores all information in the appropriate data structure and proceeds to System Feature 4.1 Audit File Output and System Feature 4.2 Display Election Information.

Response-4: A system failure occurs during User-Action-1 to System-Action-2, and the system exits with the corresponding error message.

4.7.3 Functional Requirements

REQ-1: System Feature 4.3 - Obtain File Name must function correctly and reliably.
REQ-2: System Feature 4.4 - Process CSV File must function correctly and reliably.
REQ-3: System Feature 4.5 - Resolve Ties must function correctly and reliably.
REQ-4: The system must correctly and robustly implement the largest reminder approach.

4.8 Develop Test Files

4.8.1 Description and Priority

Description: The programmers and testers create test files to reach the testing requirements and ensure the robustness of the software. The test files are utilized throughout the development process of each system feature. It is critical for programmers and testers to construct test files for the software's initial cost and risk analysis.
Priority: This feature has a risk priority of 9. It is an effective mechanism to identify bugs in the software early on in the development process and satisfy the client's requirements. It also sets up primary testing rules/scripts during the operation and maintenance step of the development cycle. Ultimately, the comprehensive implementation of this feature can prevent the cost of further revision or requirement changes.

4.8.2 Stimulus / Response Sequences

User-Action-1: The programmer or tester executes the test for the specific system feature.

Response-1: The system signals the programmers and testers the existence of bugs in the software if one or more tests fail

Response-2: The system marks all tests passed for the system feature of interest to work as intended.

4.8.3 Functional Requirements

REQ-1: System features of interest must be implemented and ready to test.
REQ-2: The system shall flag the test failure for the programmers and testers in an intuitive manner to understand the issue.
REQ-3: The system shall display the pass of the test file.

4.9 One Entry Point

4.9.1 Description and Priority

Description: The system provides one single entry point for both normal elections and special elections at any time of the year. The system is up-to-date with rules for any new political policy of special elections conducted during the year.
Priority: This feature has a cost priority of 4. It improves the user experience by providing a one-stop service without the need to switch to a different software system. It is not a critical feature of the software, but it saves the cost of developing an extra software system.

4.9.2 Stimulus / Response Sequences

User-Action-1: The user tests the system using a test file (see System Feature 4.8) or executes the system with an actual election votes file (see System Feature 4.3) during normal election time.

Response-1: The system detects the type of election given the input files as the normal election and applies its corresponding rules. The system proceeds to the next process.

User-Action-2: The user tests the system using a test file (see System Feature 4.8) or executes the system with an actual election votes file (see System Feature 4.3) during special election time.

Response-2: The system detects the type of election given the input files as a special election and applies its corresponding rules. The system proceeds to the next process.

User-Action-3: The user tests the system using a test file (see System Feature 4.8) or executes the system with an actual election votes file (see System Feature 4.3) during alternative election time.

Response-3: The system detects an alternative election type without a corresponding rule, and the system exits with an error message indicating the invalidity of the election type.

Response-4: A system failure occurs among User-Action-1, User-Action-2, and User-Action-3, and the system exits with the corresponding error message.

4.9.3 Functional Requirements

REQ-1: System Feature 4.3 - Obtain File Name shall function.
REQ-2: System Feature 4.8 - Develop Test Files shall function.
REQ-3: All relevant system features must contain rules to handle both normal elections and all special elections
REQ-4: The system shall indicate the type of election by displaying a message to the user.
REQ-5: The system shall display any error message in a manner that is intuitive to understand.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must be able to process 100,000 ballots in under four minutes. This is equivalent to 25,000 ballots per minute. The large number of ballots to be processed is needed so that counting a large number of votes can be done quickly and efficiently, as an election in most uses will have well over 100,000 ballots cast.

5.2 Safety Requirements

It is important for election systems to be well secured, so that ballots and results cannot be tampered with. If votes are manipulated in any way, then trust in the electoral system will be damaged. Therefore, there should be a way to verify votes in cases of tampering or malfunctions within the system, such as a paper trail. This way, votes can be counted manually if the system is no longer functional. The system will produce an audit file to aid in system diagnostics and verification. Should the system malfunction, it is theoretically possible that the CSV file be corrupted.

5.3 Security Requirements

It is the user's responsibility that the voting results CSV file be secured and authenticated. The system itself safely handles the file to prevent time of check / time of use errors and other race conditions. The file is also consumed and checked in a manner to prevent buffer overflows and other vulnerabilities. The length of the filename is limited to 255 bytes with a path length not exceeding 4096 bytes. The use of external libraries is minimized, with only essential environment variables being used. A cryptographically secure random library is used when simulating the coin toss to break ties, thus hindering the ability of a malicious actor to influence the final result.

5.4 Software Quality Attributes

The GUI used is intuitive, following the GUI format used in common user applications. The ReadMe is intuitive, providing useful information for future developers and informed users. The system is well documented and commented, with code that is easy to read and scalable. The code itself is testable at both algorithmic, modular, and application levels. The system is reliable and open to expansion, with replicable results. This system is designed to be easy to use with few user inputs.

5.5 Business Rules

The system is designed such that any user is able to use it. However, the collection of ballots and checks to the validity of inputted files must be determined by election officials. The input files must match the constraints of the system so the system can produce consistent and reliable results. Invalid or corrupted files will be rejected, though the validity of the submitted ballot file cannot be known by the system and must be guaranteed by the election officials.

6. Other Requirements

6.1 Legal Requirements - Open Party List Voting

Open Party List Voting allows voters to rank both party and party candidates. Winners of the election are decided by first ordering candidates by number of votes, then assigning a party seats based on quota and remainder. Quota is calculated by dividing the total number of votes by the total number of seats. Each party is given a number of seats at least equal to the integer of this quota. Party members are assigned seats by rank, as determined by the voters. Should a party have insufficient members, the remaining seats are assigned to the next ranked party. Seats remaining after the quota phase, and as a result of non-integer quotas, are assigned to parties in the remainder phase. In this phase, parties are ranked by the greatest fractional part of the quota. Seat assignment proceeds as before.

6.2 Legal Requirements - Closed Party List Voting

Closed Party List Voting is a traditional style of election voting with a fixed ordering of candidates determined by each party. Voters are informed of the preordered candidates listed for each party on the ballot but do not have the option to vote for any specific candidates; they vote for the entire party instead. Winners of the election are again decided by quota and remainder as in §6.1, with seats being assigned to candidates based on their rank as determined by the party and the party's rank as determined by the voters.

Appendix A: Glossary

Term	**Definition**
Graphical User Interface (GUI)	A type of interface that allows end-users to interact with a system through icons and graphics
Comma Separated Values (CSV)	A type of file format where tabular data is stored as comma-separated values
Open Party Listing (OPL)	A type of voting system where voters express their preference for a particular candidate
Closed Party Listing (CPL)	A type of voting system where voters express their preference for a political party/affiliation
GitHub	An online repository for version control and distribution.

Appendix B: Analysis Models

TBD-1

Appendix C: To Be Determined List

TBD-1: Appendix B: Analysis Models